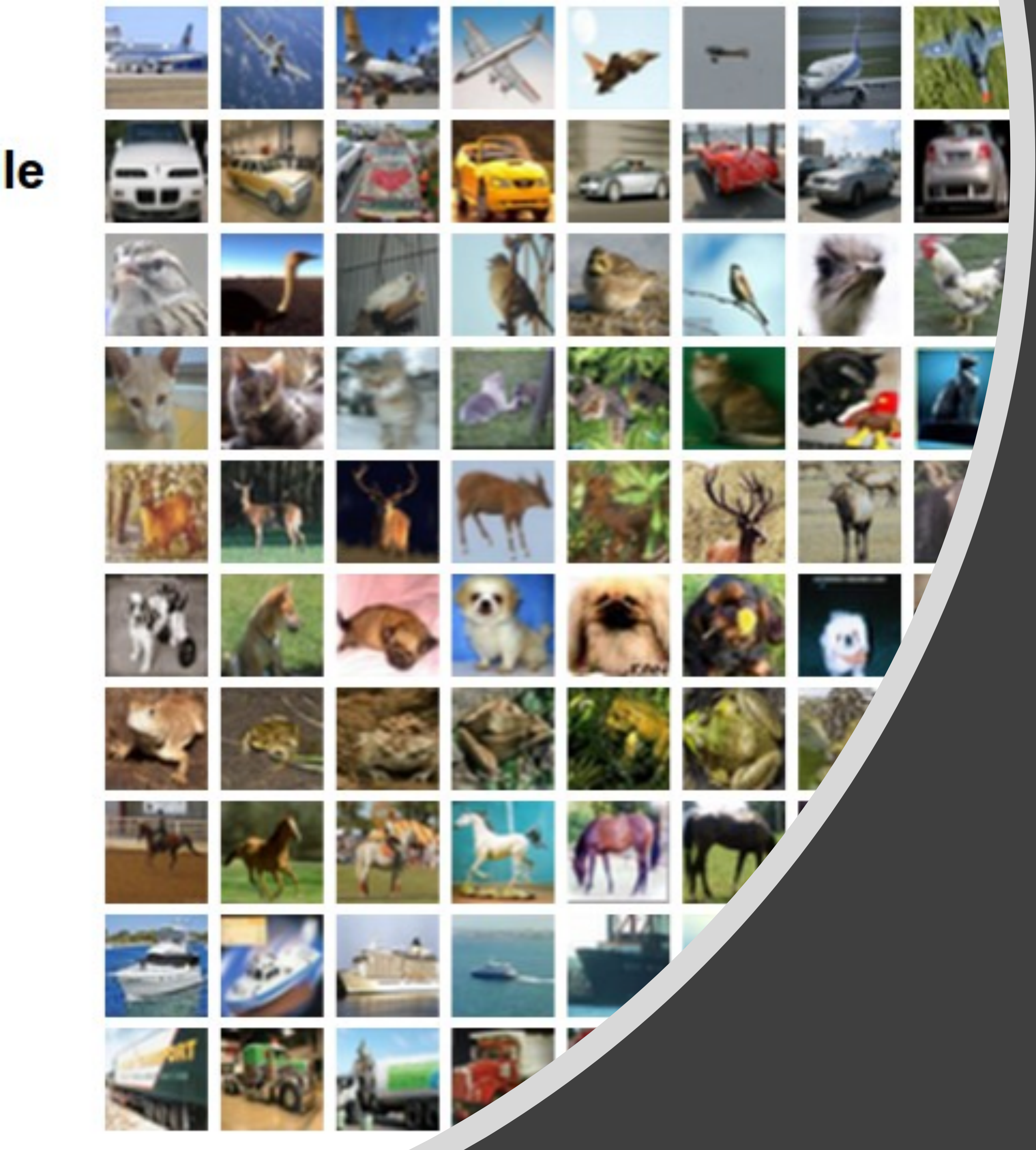
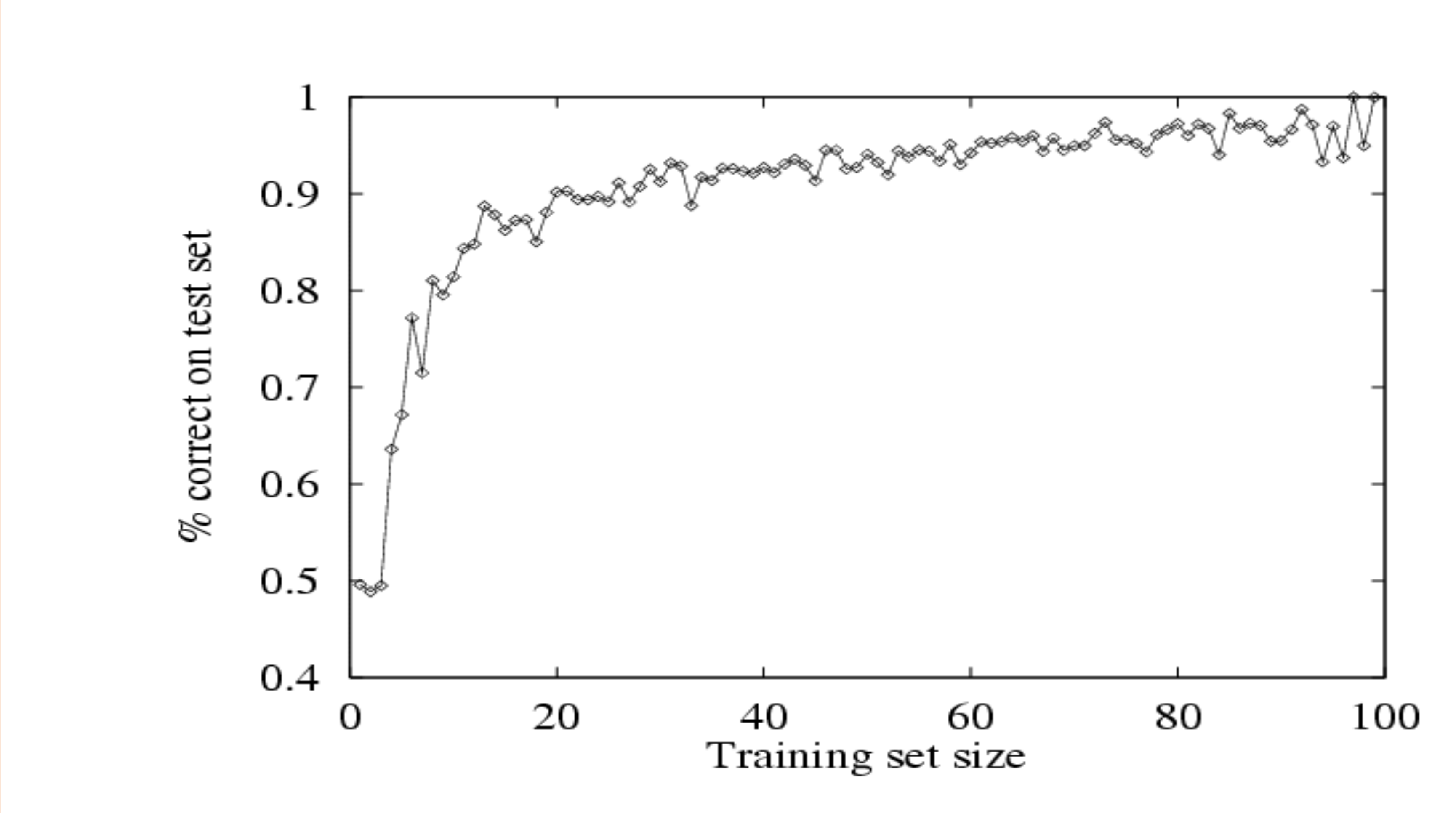


Evaluating Models

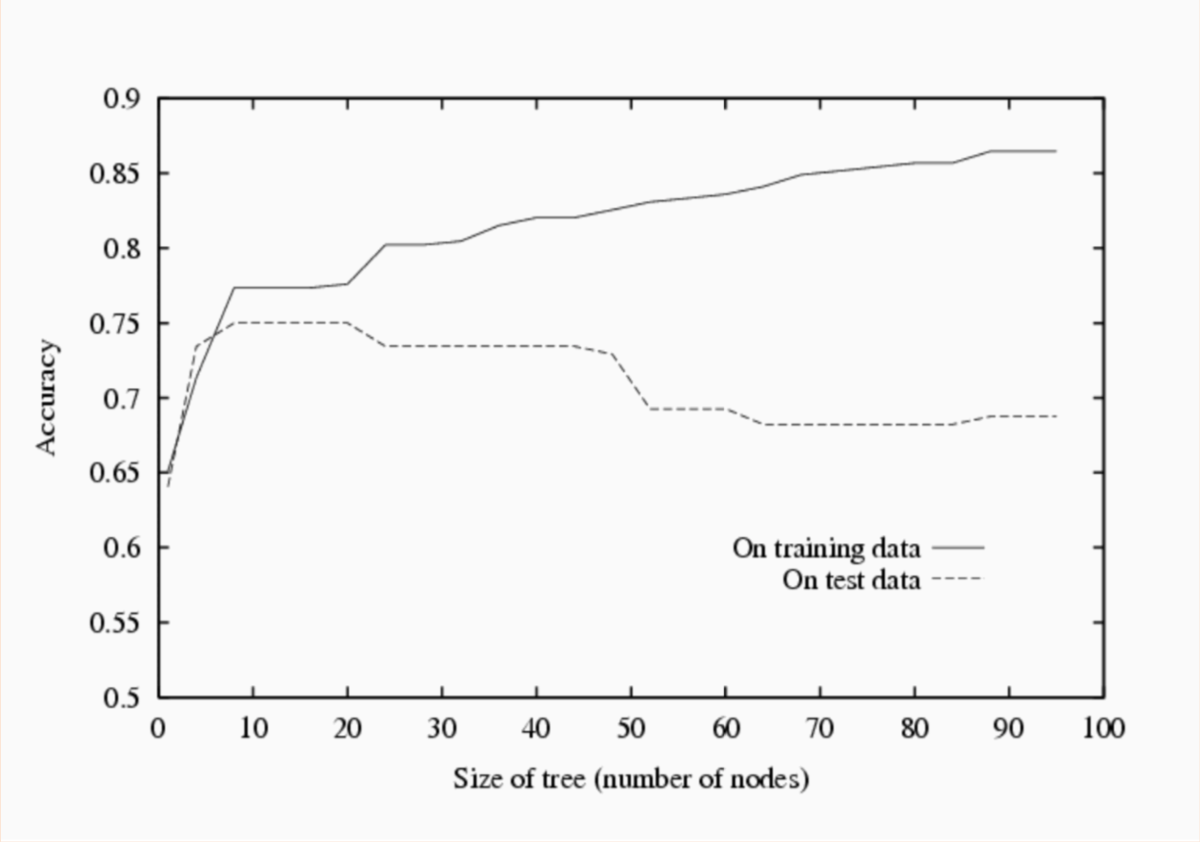
Plan for Class

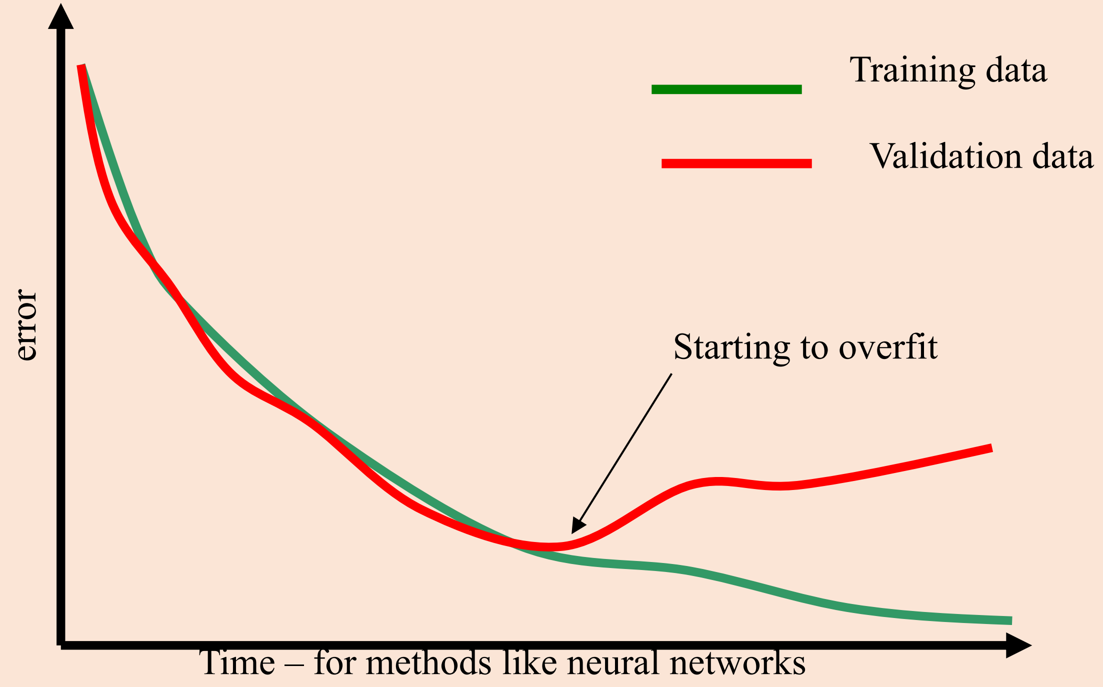
- ❑ Learning curve + overfitting recap
- ❑ Validation and cross validation
- ❑ Accuracy and beyond
- ❑ Confusion matrix
- ❑ ROC curve

Learning Curve

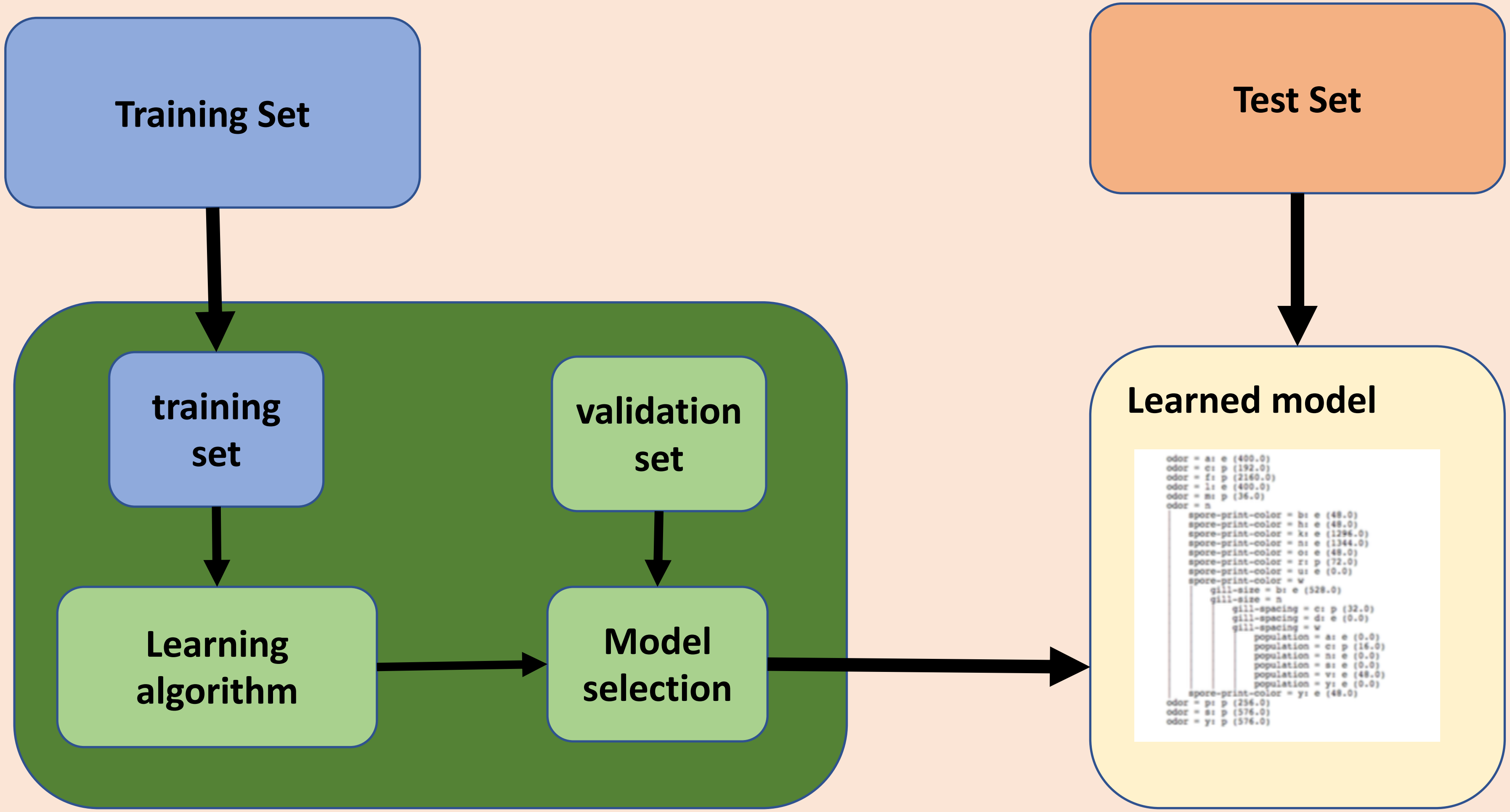


Overfitting

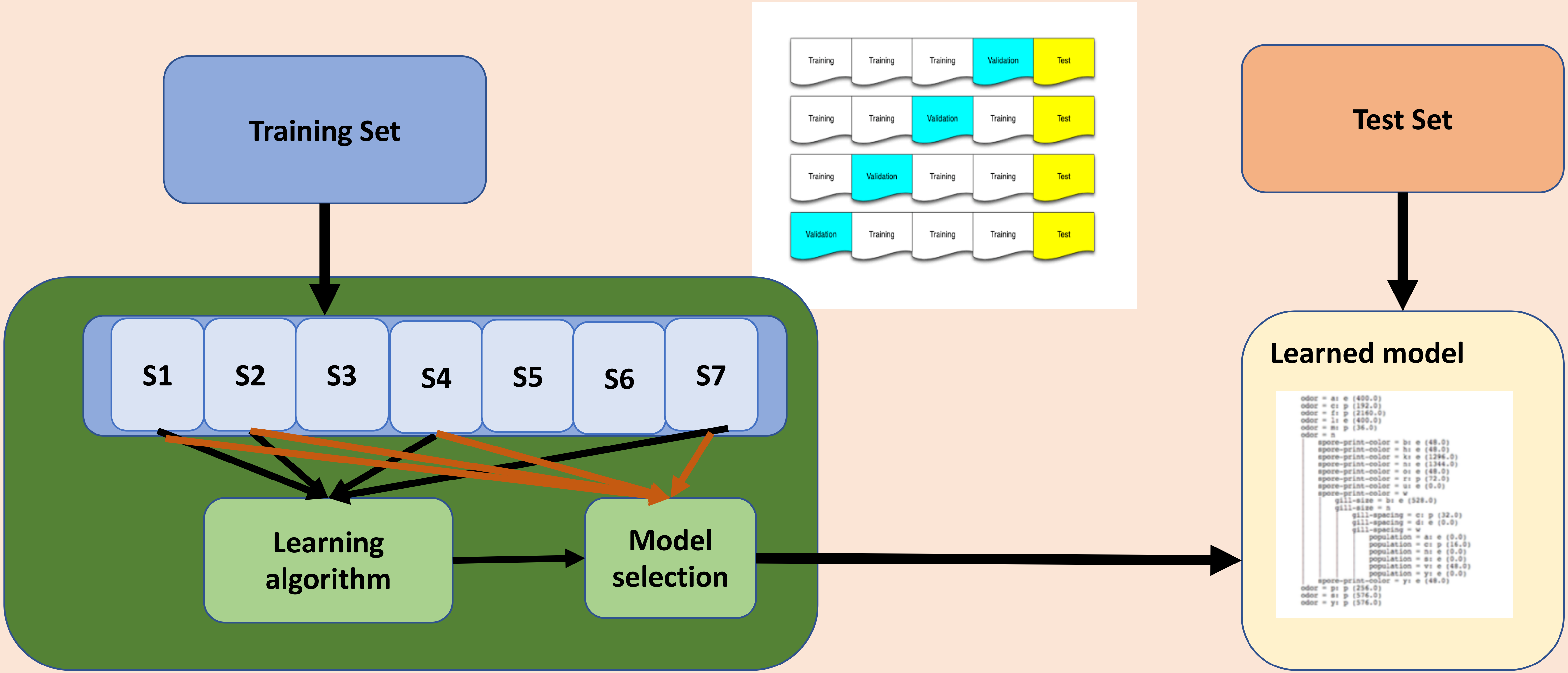




Validation


Training Set
Test Set
training set
validation set
Learning algorithm
Model selection
Learned model
odor = a: e (400.0)
odor = c: p (192.0)
odor = f: p (2160.0)
odor = l: e (400.0)
odor = m: p (36.0)
odor = n
spore-print-color = b: e (48.0)
spore-print-color = h: e (48.0)
spore-print-color = k: e (1296.0)
spore-print-color = n: e (1344.0)
spore-print-color = o: e (48.0)
spore-print-color = r: p (72.0)
spore-print-color = u: e (0.0)
spore-print-color = w
gill-size = b: e (528.0)
gill-size = n
gill-spacing = c: p (32.0)
gill-spacing = d: e (0.0)
gill-spacing = w
population = a: e (0.0)
population = c: p (16.0)
population = n: e (0.0)
population = s: e (0.0)
population = v: e (48.0)
population = y: e (0.0)
spore-print-color = y: e (48.0)
odor = p: p (256.0)
odor = s: p (576.0)
odor = y: p (576.0)

Cross Validation



Cross Validation

In [20]:
```
#Loading  train an d test data
train_set_x_orig,train_set_y,test_set_x_orig,test_set_y,classes=load_datase
#Lets get some basic data about our image numpy arrays
m_train = train_set_x_orig.shape[0]
m_test = test_set_x_orig.shape[0]
num_px = train_set_x_orig.shape[1]
print("Number of training examples: m_train = " + str(m_train))
print("Number of test examples: m_test = " + str(m_test))
print("Height/Width of each image: num_px = " + str(num_px))
print("Each image is of size: ("+ str(num_px) + ", " + str(num_px) + ", 3)"
print("train_set_x shape: " + str(train_set_x_orig.shape))
print("train_set_y shape: " + str(train_set_y.shape))
print("test_set_x shape : " + str(test_set_x_orig.shape))
print("test_set_y shape: "+ str(test_set_y.shape))
```

```
Number of training examples: m_train = 209
Number of test examples: m_test = 50
Height/Width of each image: num_px = 64
Each image is of size: (64, 64, 3)
train_set_x shape: (209, 64, 64, 3)
train_set_y shape: (1, 209)
test_set_x shape : (50, 64, 64, 3)
test_set_y shape: (1, 50)
```

Cross Validation

In [22]:
```
#  We flatten the numpy array from (num_px, num_px, 3)
#  to (num_px*num_px*3, 1) this will make it easier for us so that each
#  image in one numpy array column
train_set_x_flatten=train_set_x_orig.reshape(train_set_x_orig.shape[0],-1).T
test_set_x_flatten=test_set_x_orig.reshape(test_set_x_orig.shape[0],-1).T

print("train_set_x_flatten shape: " + str(train_set_x_flatten.shape))
print("train_set_y shape: " + str(train_set_y.shape))
print("test_set_x_flatten shape: "+ str(test_set_x_flatten.shape))
print("test_set_y shape: "+ str(test_set_y.shape))

#Standardize the dataset for images by dividing each by 255
train_set_x = train_set_x_flatten/255
test_set_x = test_set_x_flatten/255
```

```
train_set_x_flatten shape: (12288, 209)
train_set_y shape: (1, 209)
test_set_x_flatten shape: (12288, 50)
test_set_y shape: (1, 50)
```

Accuracy

So far our analysis has been focused on accuracy:

$$accuracy = \frac{\#\ correct\ classifications}{\#\ classifications}$$

Is Accuracy enough?

Image 1

Image 5

Image 17

Image 6

Image 10

Is Accuracy enough?

Not cat

Not cat

Cat

Cat

Cat

Is Accuracy enough?

Image 1

Image 15

Image 9

Image 16

Image 5

Image 11

Image 13

Image 19

Is Accuracy enough?

Not cat

Not cat

Not cat

Not cat

Not cat

cat

cat

cat

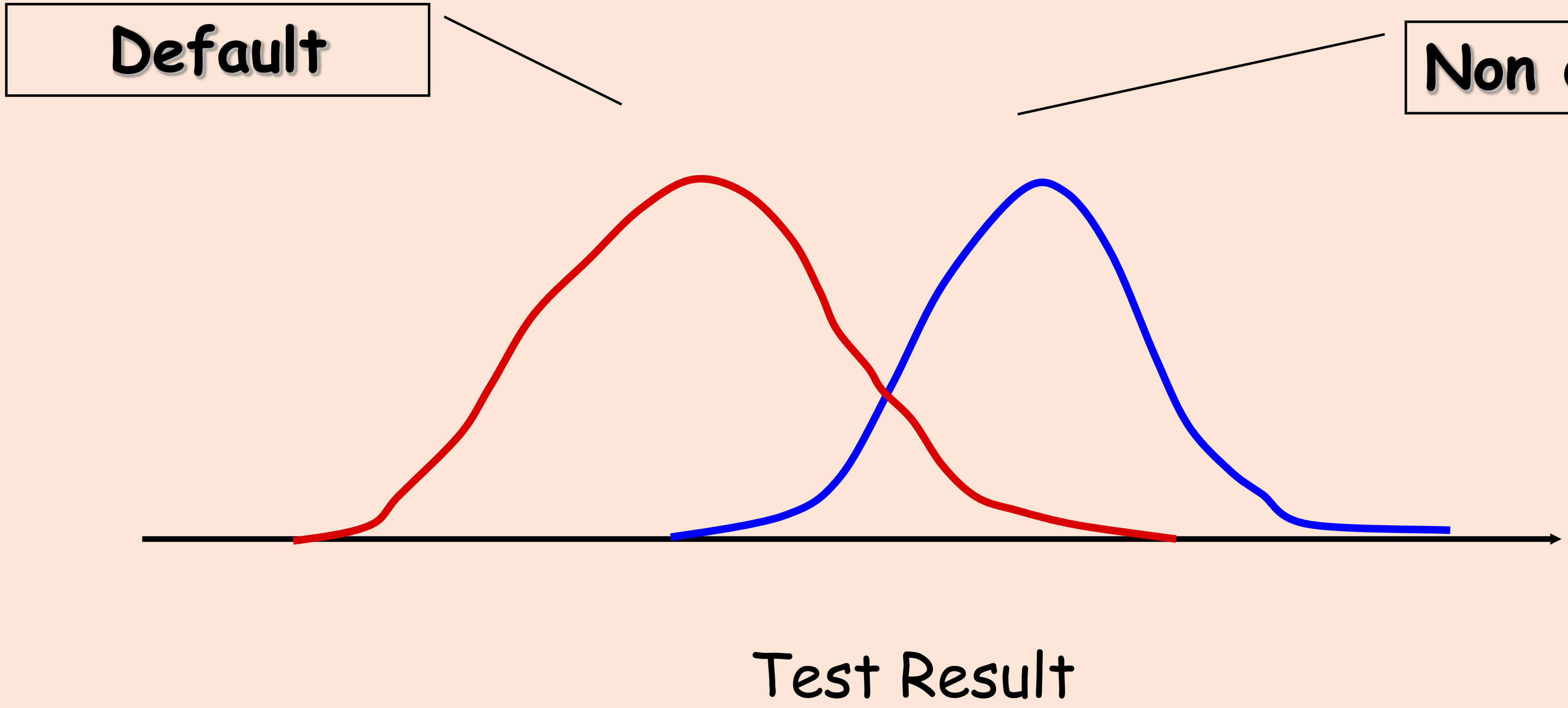
Confusion Matrix

Real life \ model	'not cat'	'cat'
Not Cat (D = 0)	☺ True negative	X Type I error (False positive)
Cat (D = 1)	X Type II error (False negative)	☺ True positive

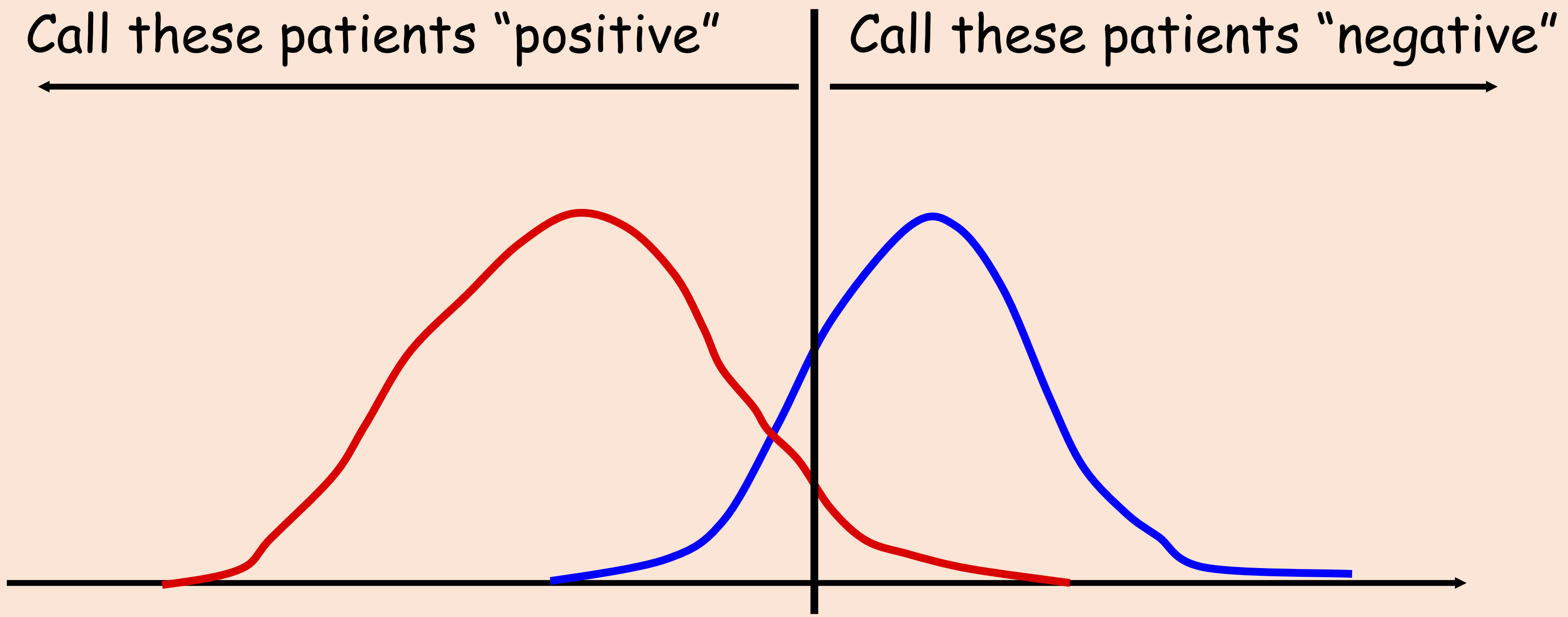
Confusion Matrix

model / Real life	'don't lend'	'lend'
Not default (D = 0)	☺ True negative	X Type I error (False positive) α
defualt (D = 1)	X Type II error (False negative) β	☺ True positive

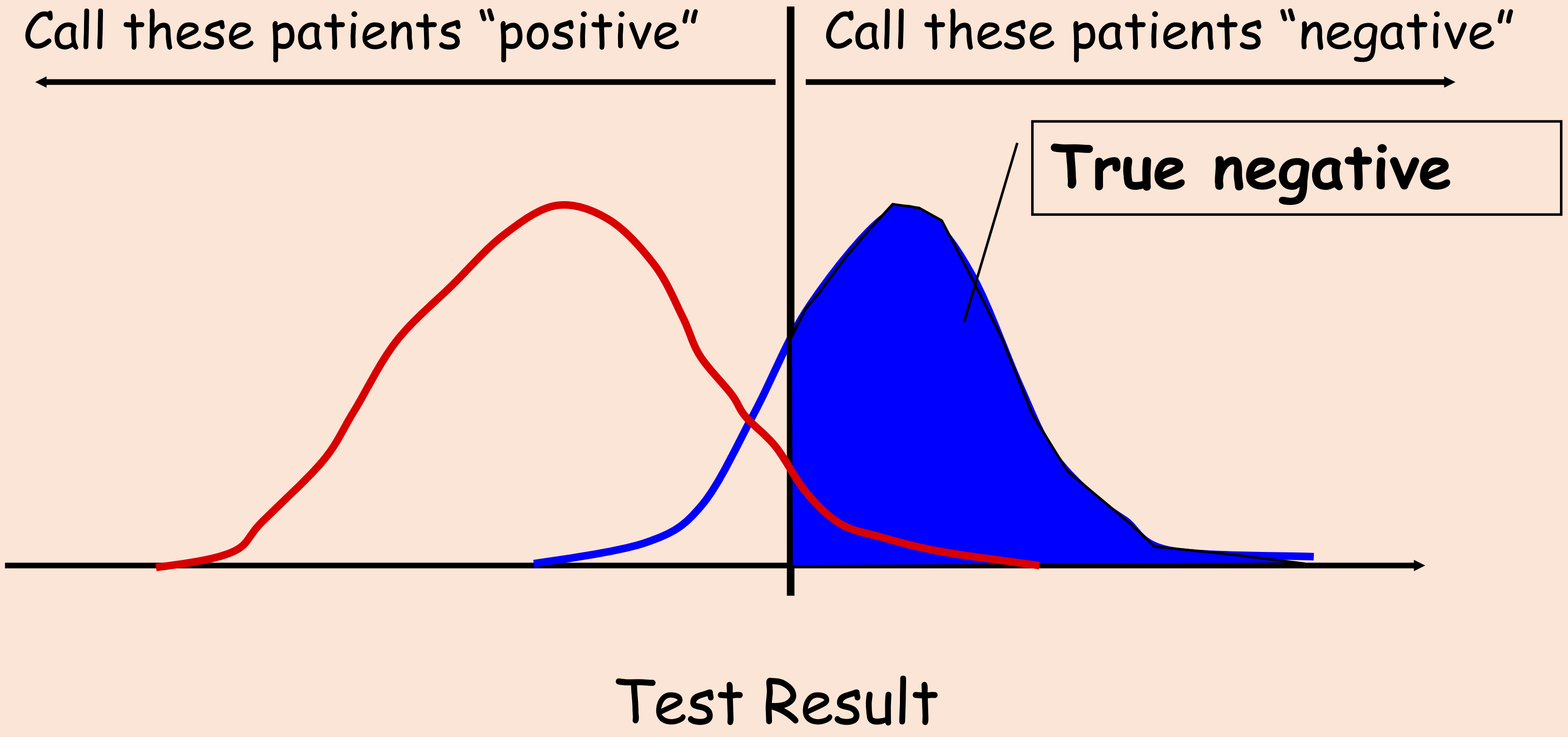
Example – Lending Club



Identifying a default



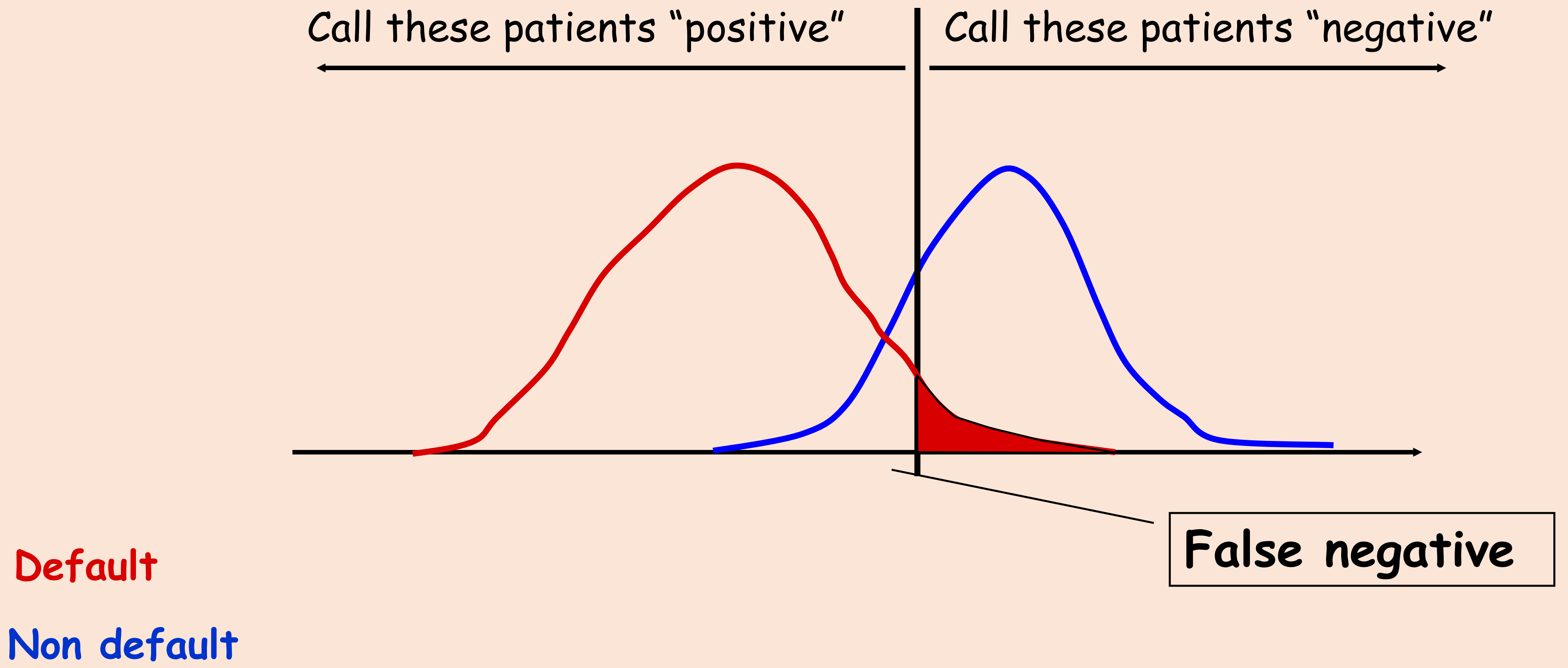
True Negative



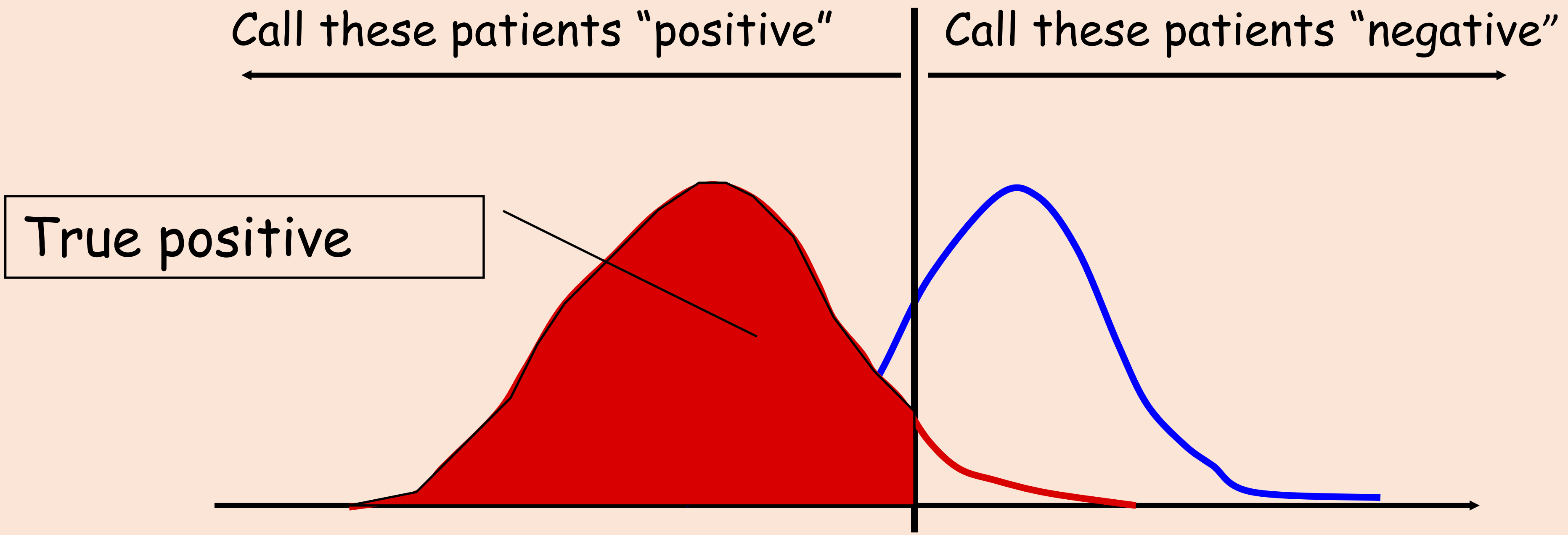
Default

Non default

False Negative



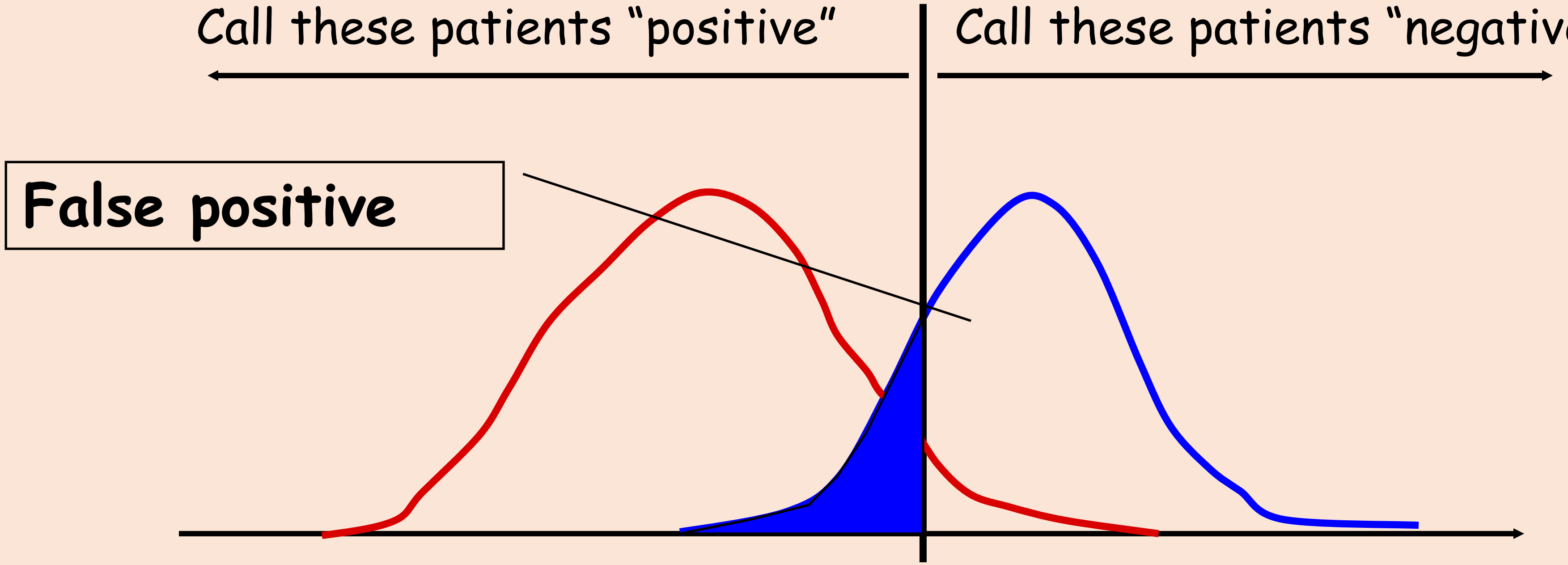
True Positive



Default

Non default

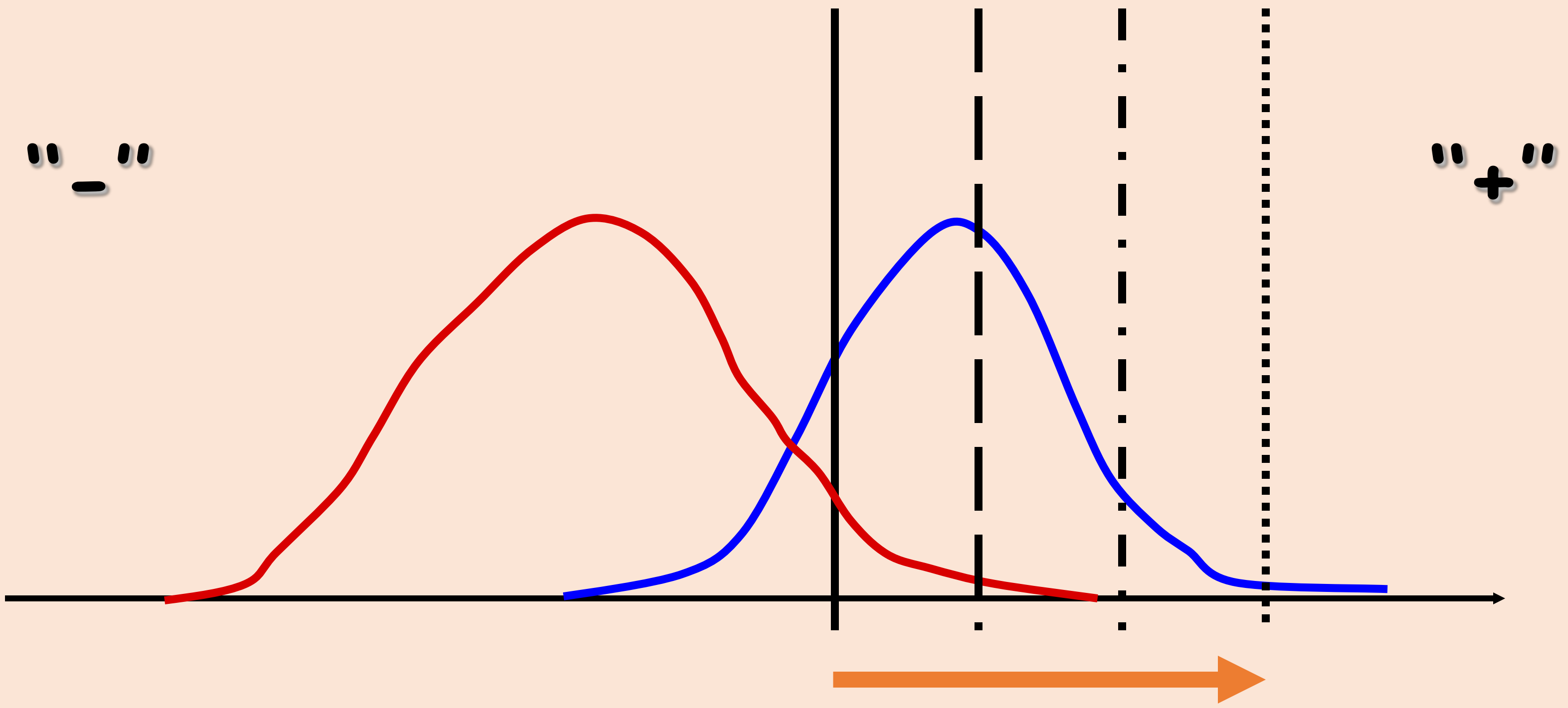
False Positive



Default

Non default

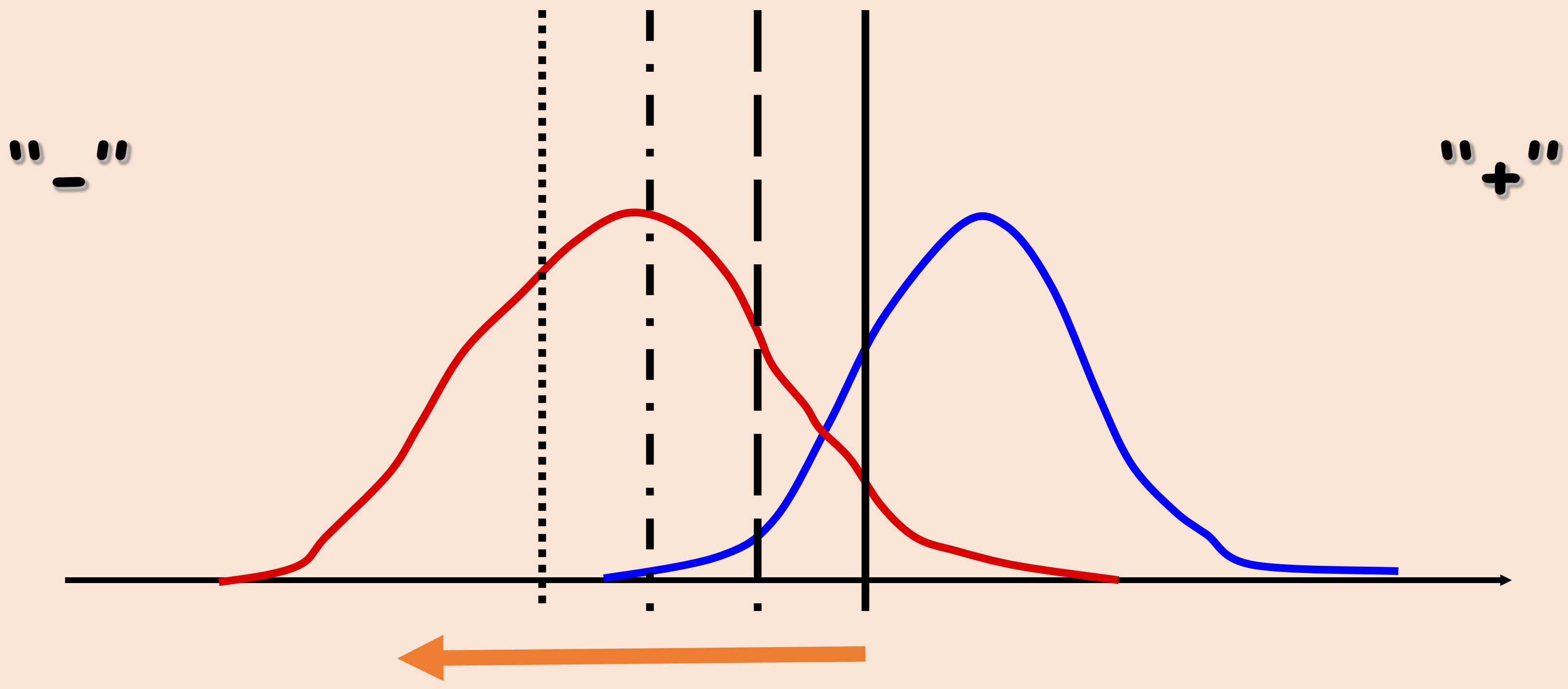
Moving the Threshold to the right



Default

Non default

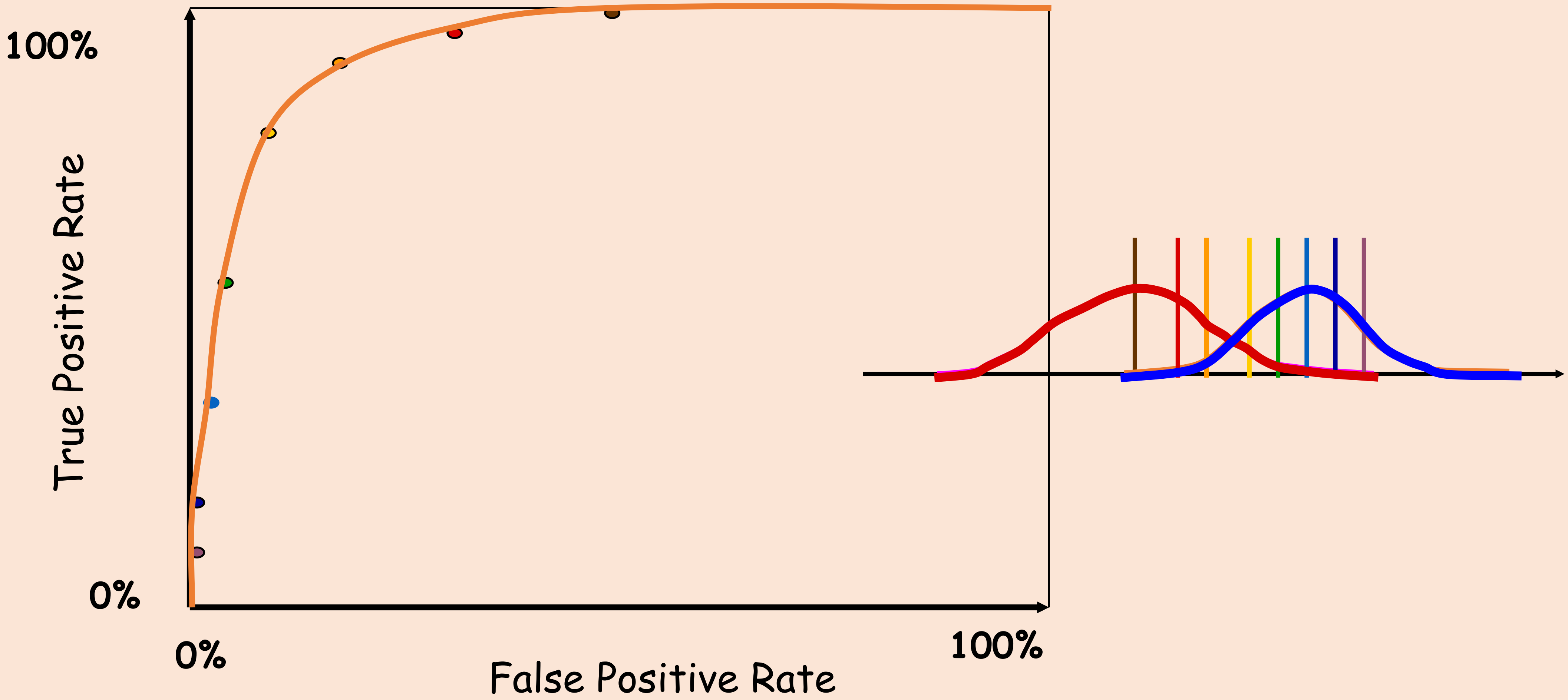
Moving the Threshold to the left



Default

Non default

ROC curve



Default

Non default

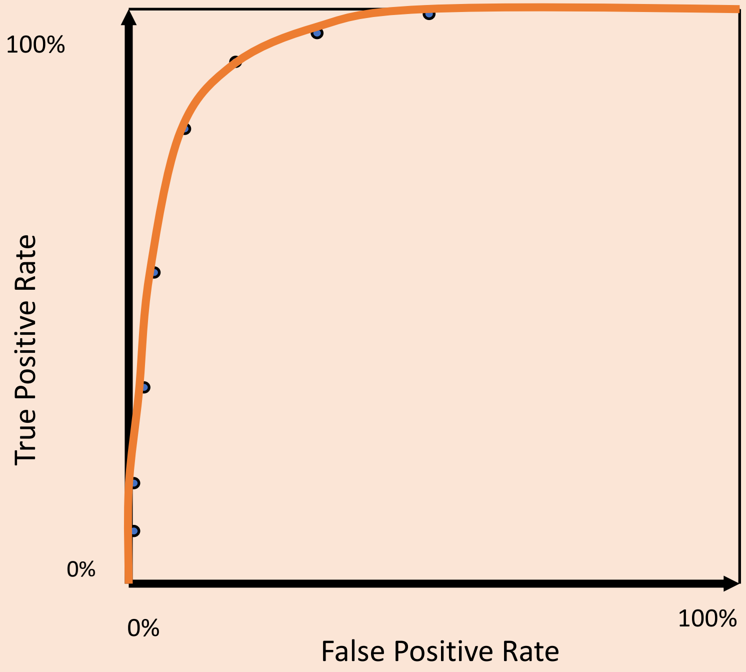
ROC analysis

ROC = Receiver Operating Characteristic

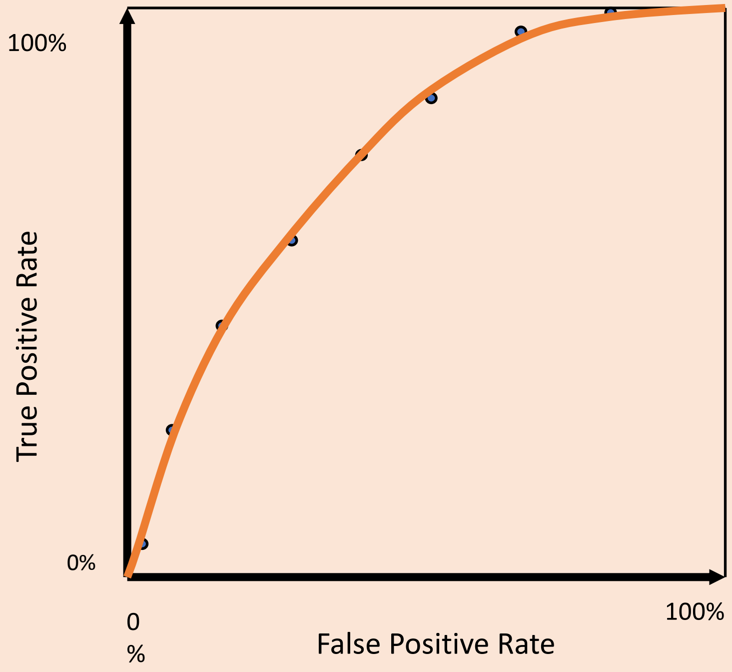
- Started in electronic signal detection theory (1940s - 1950s)
- Used extensively for radar signal analysis
- Has become very popular in biomedical applications, particularly radiology and imaging
- Used in machine learning applications to assess classifiers
- Used in many business applications
- Can be used to compare tests/procedures

ROC curve comparison

A good test:

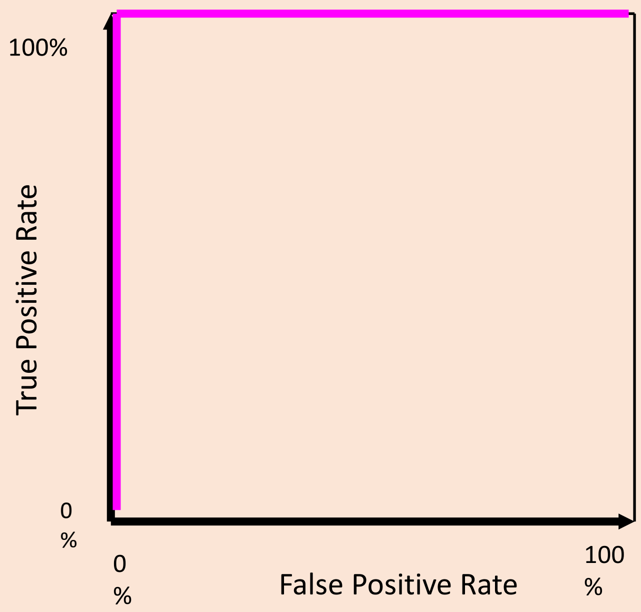


A poor test:



Confusion Matrix

Best Test:



The distributions don't overlap at all

Worst test:

The distributions overlap completely

Confusion Matrix

Default

Non default